Optimal algorithms for smooth and strongly convex distributed optimization in networks

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, Laurent Massoulié

▶ **To cite this version:**

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. 2017. hal-01478317

HAL Id: hal-01478317
https://hal.science/hal-01478317

Preprint submitted on 28 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimal algorithms for smooth and strongly convex distributed optimization in networks

Kevin Scaman[1] Francis Bach[2]
Sébastien Bubeck[3] Yin Tat Lee[3] Laurent Massoulié[1]

[1] MSR-INRIA Joint Center, Palaiseau, France
[2] INRIA, Ecole Normale Supérieure, Paris, France
[3] Theory group, Microsoft Research, Redmond, United States
{kevin.scaman,francis.bach,laurent.massoulie}@inria.fr
{yile,sebubeck}@microsoft.com

Abstract

In this paper, we determine the optimal convergence rates for strongly convex and smooth distributed optimization in two settings: centralized and decentralized communications over a network. For centralized (i.e. *master/slave*) algorithms, we show that distributing Nesterov's accelerated gradient descent is optimal and achieves a precision $\varepsilon > 0$ in time $O(\sqrt{\kappa_g}(1 + \Delta\tau)\ln(1/\varepsilon))$, where κ_g is the condition number of the (global) function to optimize, Δ is the diameter of the network, and τ (resp. 1) is the time needed to communicate values between two neighbors (resp. perform local computations). For decentralized algorithms based on gossip, we provide the first optimal algorithm, called the *multi-step dual accelerated* (MSDA) method, that achieves a precision $\varepsilon > 0$ in time $O(\sqrt{\kappa_l}(1 + \frac{\tau}{\sqrt{\gamma}})\ln(1/\varepsilon))$, where κ_l is the condition number of the local functions and γ is the (normalized) eigengap of the gossip matrix used for communication between nodes. We then verify the efficiency of MSDA against state-of-the-art methods for two problems: least-squares regression and classification by logistic regression.

1 Introduction

Given the numerous applications of distributed optimization in machine learning, many algorithms have recently emerged, that allow the minimization of objective functions f defined as the average $\frac{1}{n}\sum_{i=1}^n f_i$ of functions f_i which are respectively accessible by separate nodes in a network [1, 2, 3, 4]. These algorithms typically alternate local incremental improvement steps (such as gradient steps) with communication steps between nodes in the network, and come with a variety of convergence rates (see for example [5, 4, 6, 7]).

Two main regimes have been looked at: (a) *centralized* where communications are precisely scheduled and (b) *decentralized* where communications may not exhibit a precise schedule. In this paper, we consider these two regimes for objective functions which are smooth and strongly-convex and for which algorithms are linearly (exponentially) convergent. The main contribution of this paper

is to propose new and matching upper and lower bounds of complexity for this class of distributed problems.

The optimal complexity bounds depend on natural quantities in optimization and network theory. Indeed, (a) for a single machine the optimal number of gradient steps to optimize a function is proportional to the square root of the condition number [8], and (b) for mean estimation, the optimal number of communication steps is proportional to the diameter of the network in centralized problems or to the square root of the eigengap of the Laplacian matrix in decentralized problems [9]. As shown in Section 3, our lower complexity bounds happen to be combinations of the two contributions above.

These lower complexity bounds are attained by two separate algorithms. In the centralized case, the trivial distribution of Nesterov's accelerated gradient attains this rate, while in the decentralized case, as shown in Section 4, the rate is achieved by a dual algorithm. We compare favorably our new optimal algorithms to existing work in Section 5.

Related work. Decentralized optimization has been extensively studied and early methods such as decentralized gradient descent [1, 10] or decentralized dual averaging [3] exhibited sublinear convergence rates. More recently, a number of methods with provable linear convergence rates were developed, including EXTRA [4, 11], augmented Lagrangians [6], and more recent approaches [7]. The most popular of such approaches is the distributed alternating direction method of multipliers (D-ADMM) [2, 12, 5] and has led to a large number of variations and extensions. In a different direction, second order methods were also investigated [13, 14]. However, to the best of our knowledge, the field still lacks a coherent theoretical understanding of the optimal convergence rates and its dependency on the characteristics of the communication network. In several related fields, complexity lower bounds were recently investigated, including the sequential optimization of a sum of functions [15, 16], distributed optimization in flat (i.e. totally connected) networks [17, 18], or distributed stochastic optimization [19].

2 Distributed optimization setting

2.1 Optimization problem

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a connected simple (i.e. undirected) graph of n computing units and diameter Δ, each having access to a function $f_i(\theta)$ over $\theta \in \mathbb{R}^d$. We consider minimizing the average of the local functions

$$\min_{\theta \in \mathbb{R}^d} \bar{f}(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \tag{1}$$

in a distributed setting. More specifically, we assume that:

1. Each computing unit can compute first-order characteristics, such as the gradient of its own function or its Fenchel conjugate. By renormalization of the time axis, and without loss of generality, we assume that this computation is performed in one unit of time.
2. Each computing unit can communicate values (i.e. vectors in $\mathbb{R}^d$) to its neighbors. This communication requires a time τ (which may be smaller or greater than 1).

These actions may be performed asynchronously and in parallel, and each node i possesses a local version of the parameter, which we refer to as θ_i. Moreover, we assume that each function f_i is α-strongly convex and β-smooth, and we denote by $\kappa_l = \frac{\beta}{\alpha} \geq 1$ the local condition number. We also denote by α_g, β_g and κ_g, respectively, the strong convexity, smoothness and condition number of the average (global) function $\bar{f}$. Note that we always have $\kappa_g \leq \kappa_l$, while the opposite inequality is, in general, not true (take for example $f_1(\theta) = \mathbb{1}\{\theta < 0\}\theta^2$ and $f_2(\theta) = \mathbb{1}\{\theta > 0\}\theta^2$ for which $\kappa_l = +\infty$ and $\kappa_g = 1$). However, the two quantities are close (resp. equal) when the local functions are similar (resp. equal) to one another.

2.2 Decentralized communication

A large body of literature considers a decentralized approach to distributed optimization based on the *gossip* algorithm [9, 1, 3, 12]. In such a case, communication is represented as a matrix multiplication with a matrix W verifying the following constraints:

1. W is an $n \times n$ symmetric matrix,
2. W is positive semi-definite,
3. The kernel of W is the set of constant vectors: $\mathrm{Ker}(W) = \mathrm{Span}(\mathbb{1})$, where $\mathbb{1} = (1, ..., 1)^\top$,
4. W is defined on the edges of the network: $W_{ij} \neq 0$ only if $i = j$ or $(i, j) \in \mathcal{E}$.

The third condition will ensure that the gossip step converges to the average of all the vectors shared between the nodes. We will denote the matrix W as the *gossip matrix*, since each communication step will be represented using it. Note that a simple choice for the gossip matrix is the Laplacian matrix $L = D - A$, where A is the adjacency matrix of the network and $D = \mathrm{diag}\left(\sum_i A_{ij}\right)$. However, in the presence of large degree nodes, weighted Laplacian matrices are usually a better choice, and the problem of optimizing these weights is known as the fastest distributed consensus averaging problem and is investigated by [20, 21].

We will denote by $\lambda_1(W) \geq \cdots \geq \lambda_n(W) = 0$ the spectrum of the gossip matrix W, and its (normalized) *eigengap* the ratio $\gamma(W) = \lambda_{n-1}(W)/\lambda_1(W)$ between the second smallest and the largest eigenvalue. Equivalently, this is the inverse of the condition number of W projected on the space orthogonal to the constant vector $\mathbb{1}$. This quantity will be the main parameter describing the connectivity of the communication network in Section 3.3 and Section 4.

3 Optimal convergence rates

In this section, we prove oracle complexity lower bounds for distributed optimization in two settings: strongly convex and smooth functions for centralized (i.e. master/slave) and decentralized algorithms based on a gossip matrix W.

In the first setting, we show that distributing accelerated gradient descent matches the optimal convergence rate, while, in the second setting, the algorithm proposed in Section 4 is shown to be optimal. Note that we will use the notation $g(\varepsilon) = \Omega(f(\varepsilon))$ for $\exists C > 0$ s.t. $\forall \varepsilon > 0, g(\varepsilon) \geq Cf(\varepsilon)$, and will, for simplicity, omit the additive terms that do not depend on the precision ε in Corollary 1 and Corollary 2.

3.1 Black-box optimization procedures

The lower bounds provided hereafter depend on a new notion of black-box optimization procedures for the problem in Eq. (1), where we consider distributed algorithms verifying the following constraints:

1. **Local memory:** each node i can store past values in a (finite) internal memory $\mathcal{M}_{i,t} \subset \mathbb{R}^d$ at time $t \geq 0$. These values can be accessed and used at time t by the algorithm run by node i, and are updated either by local computation or by communication (defined below), that is, for all $i \in \{1, ..., n\}$,
$$\mathcal{M}_{i,t} \subset \mathcal{M}_{i,t}^{comp} \cup \mathcal{M}_{i,t}^{comm}. \tag{2}$$

2. **Local computation:** each node i can, at time t, compute the gradient of its local function $\nabla f_i(\theta)$ or its Fenchel conjugate $\nabla f_i^*(\theta)$ for a value $\theta \in \mathcal{M}_{i,t}$ in the node's internal memory, that is, for all $i \in \{1, ..., n\}$,
$$\mathcal{M}_{i,t}^{comp} = \mathrm{Span}\left(\{\theta, \nabla f_i(\theta), \nabla f_i^*(\theta) : \theta \in \mathcal{M}_{i,t-1}\}\right). \tag{3}$$

3. **Local communication:** each node i can, at time t, share a value to all or part of its neighbors, that is, for all $i \in \{1, ..., n\}$,
$$\mathcal{M}_{i,t}^{comm} = \mathrm{Span}\Big(\bigcup_{(i,j)\in\mathcal{E}} \mathcal{M}_{j,t-\tau} \Big). \tag{4}$$

4. **Output value:** each node i must, at time t, specify one vector in its memory as local output of the algorithm, that is, for all $i \in \{1, ..., n\}$,
$$\theta_{i,t} \in \mathcal{M}_{i,t}. \tag{5}$$

Hence, a black-box procedure will return n output values—one for each node of the network—and our analysis will focus on ensuring that *all local output values* are converging to the optimal parameter of Eq. (1). Moreover, we will say that a black-box procedure *uses a gossip matrix* W if the local communication is achieved by multiplication of a vector with W. For simplicity, we assume that all nodes start with the simple internal memory $\mathcal{M}_{i,0} = \{0\}$. Note that communications and local computations may be performed in parallel and asynchronously.

3.2 Centralized algorithms

In this section, we show that, for any black-box optimization procedure, at least $\Omega(\sqrt{\kappa_g}\ln(1/\varepsilon))$ gradient steps and $\Omega(\Delta\sqrt{\kappa_g}\ln(1/\varepsilon))$ communication steps are necessary to achieve a precision $\varepsilon > 0$, where κ_g is the global condition number and Δ is the diameter of the network. These lower bounds extend the communication complexity lower bounds for totally connected communication networks of [18], and are natural since at least $\Omega(\sqrt{\kappa_g}\ln(1/\varepsilon))$ steps are necessary to solve a strongly convex and smooth problem up to a fixed precision, and at least Δ communication steps are required to transmit a message between any given pair of nodes.

In order to simplify the proofs of the following theorems, and following the approach of [22], we will consider the limiting situation $d \to +\infty$. More specifically, we now assume that we are working in $\ell_2 = \{\theta = (\theta_k)_{k\in\mathbb{N}} \ : \ \sum_k \theta_k^2 < +\infty\}$ rather than $\mathbb{R}^d$.

Theorem 1. *Let $\mathcal{G}$ be a graph of diameter $\Delta > 0$ and size $n > 0$, and $\beta_g \geq \alpha_g > 0$. There exists n functions $f_i : \ell_2 \to \mathbb{R}$ such that $\bar{f}$ is α_g strongly convex and β_g smooth, and for any $t \geq 0$ and any black-box procedure one has, for all $i \in \{1, ..., n\}$,*

$$\bar{f}(\theta_{i,t}) - \bar{f}(\theta^*) \geq \frac{\alpha_g}{2}\left(1 - \frac{4}{\sqrt{\kappa_g}}\right)^{1+\frac{t}{1+\Delta\tau}} \|\theta_{i,0} - \theta^*\|^2, \tag{6}$$

where $\kappa_g = \beta_g/\alpha_g$.

The proof of Theorem 1 relies on splitting the function used by Nesterov to prove oracle complexities for strongly convex and smooth optimization [8, 22] on two nodes at distance Δ. One can show that most dimensions of the parameters $\theta_{i,t}$ will remain zero, and local gradient computations may only increase the number of non-zero dimensions by one. Finally, at least Δ communication rounds are necessary in-between every gradient computation, in order to share information between the two nodes. The detailed proof is available as supplementary material.

Corollary 1. *For any graph of diameter Δ and any black-box procedure, there exists functions f_i such that the time to reach a precision $\varepsilon > 0$ is lower bounded by*

$$\Omega\left(\sqrt{\kappa_g}\Big(1 + \Delta\tau\Big)\ln\left(\frac{1}{\varepsilon}\right)\right), \tag{7}$$

This optimal convergence rate is achieved by distributing Nesterov's accelerated gradient descent on the global function. Computing the gradient of $\bar{f}$ is performed by sending all the local gradients ∇f_i to a single node (denoted as *master node*) in Δ communication steps (which may involve several simultaneous messages), and then returning the new parameter θ_{t+1} to every node in the network (which requires another Δ communication steps). In practice, summing the gradients can be distributed by computing a spanning tree (with the root as master node), and asking for each node to perform the sum of its children's gradients before sending it to its parent. Standard methods as described by [23] can be used for performing this parallelization of gradient computations.

This algorithm has three limitations: first, the algorithm is not robust to machine failures, and the central role played by the master node also means that a failure of this particular machine may completely freeze the procedure. Second, and more generally, the algorithm requires precomputing a spanning tree, and is thus not suited to time-varying graphs, in which the connectivity between the nodes may change through time (e.g. in *peer-to-peer networks*). Finally, the algorithm requires *every* node to complete its gradient computation before aggregating them on the master node, and the efficiency of the algorithm thus depends on the slowest of all machines. Hence, in the presence of non-uniform latency of the local computations, or the slow down of a specific machine due to a hardware failure, the algorithm will suffer a significant drop in performance.

3.3 Decentralized algorithms

The gossip algorithm [9] is a standard method for averaging values across a network when its connectivity may vary through time. This approach was shown to be robust against machine failures, non-uniform latencies and asynchronous or time-varying graphs, and a large body of literature extended this algorithm to distributed optimization [1, 3, 12, 4, 6, 7, 13].

The convergence analysis of decentralized algorithms usually relies on the spectrum of the *gossip matrix* W used for communicating values in the network, and more specifically on the ratio between the second smallest and the largest eigenvalue of W, denoted γ. In this section, we show that, with respect to this quantity and κ_l, reaching a precision ε requires at least $\Omega(\sqrt{\kappa_l}\ln(1/\varepsilon))$ gradient steps and $\Omega\left(\sqrt{\frac{\kappa_l}{\gamma}}\ln(1/\varepsilon)\right)$ communication steps, by exhibiting a gossip matrix such that a corresponding lower bound exists.

Theorem 2. *Let $\alpha, \beta > 0$ and $\gamma \in (0, 1]$. There exists a gossip matrix W of eigengap $\gamma(W) = \gamma$, and α-strongly convex and β-smooth functions $f_i : \ell_2 \to \mathbb{R}$ such that, for any $t \geq 0$ and any black-box procedure using W one has, for all $i \in \{1, ..., n\}$,*

$$\bar{f}(\theta_{i,t}) - \bar{f}(\theta^*) \geq \frac{3\alpha}{2}\left(1 - \frac{16}{\sqrt{\kappa_l}}\right)^{1+\frac{t}{1+\frac{t}{5\sqrt{\gamma}}}} \|\theta_{i,0} - \theta^*\|^2, \tag{8}$$

where $\kappa_l = \beta/\alpha$ is the local condition number.

The proof of Theorem 2 relies on the same technique as that of Theorem 1, except that we now split the two functions on a subset of a linear graph. These networks have the appreciable property that $\Delta \approx 1/\sqrt{\gamma}$, and we can thus use a slightly extended version of Theorem 1 to derive the desired result. The complete proof is available as supplementary material.

Corollary 2. *For any $\gamma > 0$, there exists a gossip matrix W of eigengap γ and α-strongly convex, β-smooth functions such that, with $\kappa_l = \beta/\alpha$, for any black-box procedure using W the time to reach a precision $\varepsilon > 0$ is lower bounded by*

$$\Omega\left(\sqrt{\kappa_l}\left(1 + \frac{\tau}{\sqrt{\gamma}}\right)\ln\left(\frac{1}{\varepsilon}\right)\right). \tag{9}$$

We will see in the next section that this lower bound is met for a novel decentralized algorithm called *multi-step dual accelerated* (MSDA) and based on the dual formulation of the optimization problem. Note that these results provide optimal convergence rates with respect to κ_l and γ, but do not imply that γ is the right quantity to consider on general graphs. The quantity $1/\sqrt{\gamma}$ may indeed be very large compared to Δ, for example for star networks, for which $\Delta = 2$ and $1/\sqrt{\gamma} = \sqrt{n}$. However, on many simple networks, the diameter Δ and the eigengap of the Laplacian matrix are tightly connected, and $\Delta \approx 1/\sqrt{\gamma}$. For example, for linear graphs, $\Delta = n - 1$ and $1/\sqrt{\gamma} \approx 2n/\pi$, for totally connected networks, $\Delta = 1$ and $1/\sqrt{\gamma} = 1$, and for regular networks, $1/\sqrt{\gamma} \geq \frac{\Delta}{2\sqrt{2}\ln_2 n}$ [24]. Finally, note that the case of totally connected networks corresponds to a previous complexity lower bound on communications proven by [18], and is equivalent to our result for centralized algorithms with $\Delta = 1$.

4 Optimal decentralized algorithms

In this section, we present a simple framework for solving the optimization problem in Eq. (1) in a decentralized setting, from which we will derive several variants, including a synchronized algorithm whose convergence rate matches the lower bound in Corollary 2 . Note that the naive approach of distributing each (accelerated) gradient step by gossiping does not lead to a linear convergence rate, as the number of gossip steps has to increase with the number of iterations to ensure the linear rate is preserved. We begin with the simplest form of the algorithm, before extending it to more advanced scenarios.

Algorithm 1 Single-Step Dual Accelerated method

Input: number of iterations $T > 0$, gossip matrix $W \in \mathbb{R}^{n\times n}$, $\eta = \frac{\alpha}{\lambda_1(W)}$, $\mu = \frac{\sqrt{\kappa_l}-\sqrt{\gamma}}{\sqrt{\kappa_l}+\sqrt{\gamma}}$

Output: $\theta_{i,T}$, for $i = 1, ..., n$

$x_0 = 0$, $y_0 = 0$

for $t = 0$ to $T - 1$ **do**

  $\theta_{i,t} = \nabla f_i^*(x_{i,t})$, for all $i = 1, ..., n$

  $y_{t+1} = x_t - \eta\Theta_t W$

  $x_{t+1} = (1 + \mu)y_{t+1} - \mu y_t$

end for

4.1 Single-Step Dual Accelerated method

A standard approach for solving Eq. (1) (see [2, 6]) consists in rewriting the optimization problem as

$$\min_{\theta\in\mathbb{R}^d} \bar{f}(\theta) = \min_{\theta_1=\cdots=\theta_n} \frac{1}{n}\sum_{i=1}^n f_i(\theta_i). \tag{10}$$

Furthermore, the equality constraint $\theta_1 = \cdots = \theta_n$ is equivalent to $\Theta\sqrt{W} = 0$, where $\Theta = (\theta_1, \ldots, \theta_n)$ and W is a gossip matrix verifying the assumptions described in Section 2. Note that, since W is positive semi-definite, $\sqrt{W}$ exists and is defined as $\sqrt{W} = V^\top \Sigma^{1/2} V$, where $W = V^\top \Sigma V$ is the singular value decomposition of W. The equality $\Theta\sqrt{W} = 0$ implies that each row of Θ is constant (since $\mathrm{Ker}(\sqrt{W}) = \mathrm{Span}(\mathbb{1})$), and is thus equivalent to $\theta_1 = \cdots = \theta_n$. This leads to the following *primal version* of the optimization problem:

$$\min_{\Theta\in\mathbb{R}^{d\times n}\,:\,\Theta\sqrt{W}=0} F(\Theta), \tag{11}$$

where $F(\Theta) = \sum_{i=1}^n f_i(\theta_i)$. Since Eq. (11) is a convex problem, it is equivalent to its *dual optimization problem*:

$$\max_{\lambda\in\mathbb{R}^{d\times n}} -F^*(\lambda\sqrt{W}), \tag{12}$$

where $F^*(y) = \sup_{x\in\mathbb{R}^{d\times n}} \langle y, x\rangle - F(x)$ is the Fenchel conjugate of F, and $\langle y, x\rangle = \mathrm{tr}(y^\top x)$ is the standard scalar product between matrices.

The optimization problem in Eq. (12) is unconstrained and convex, and can thus be solved using a variety of convex optimization techniques. The proposed *single-step dual accelerated* (SSDA) algorithm described in Alg. (1) uses Nesterov's accelerated gradient descent, and can be thought of as an accelerated version of the distributed augmented Lagrangian method of [6] for $\rho = 0$. The algorithm is derived by noting that a gradient step of size $\eta > 0$ for Eq. (12) is

$$\lambda_{t+1} = \lambda_t - \eta\nabla F^*(\lambda_t\sqrt{W})\sqrt{W}, \tag{13}$$

and the change of variable $y_t = \lambda_t\sqrt{W}$ leads to

$$y_{t+1} = y_t - \eta\nabla F^*(y_t)W. \tag{14}$$

This equation can be interpreted as gossiping the gradients of the local conjugate functions $\nabla f_i^*(y_{i,t})$, since $\nabla F^*(y_t)_{ij} = \nabla f_j^*(y_{j,t})_i$.

Theorem 3. *The iterative scheme in Alg. (1) converges to $\Theta = \theta^* \mathbb{1}^\top$ where θ^* is the solution of Eq. (1). Furthermore, the time needed for this algorithm to reach any given precision $\varepsilon > 0$ is*

$$O\left((1+\tau)\sqrt{\frac{\kappa_l}{\gamma}}\ln\left(\frac{1}{\varepsilon}\right)\right). \tag{15}$$

This theorem relies on proving that the condition number of the dual objective function is upper bounded by $\frac{\kappa_l}{\gamma}$, and noting that the convergence rate for accelerated gradient descent depends on the square root of the condition number (see, e.g., [22]). A detailed proof is available as supplementary material.

4.2 Multi-Step Dual Accelerated method

The main problem of Alg. (1) is that it always performs the same number of gradient and gossip steps. When communication is cheap compared to local computations ($\tau \ll 1$), it would be preferable to perform more gossip steps than gradient steps in order to propagate the local gradients further than the local neighborhoods of each node. This can be achieved by replacing W by $P_K(W)$ in Alg. (1), where P_K is a polynomial of degree at most K. If $P_K(W)$ is itself a gossip matrix, then the analysis of the previous section can be applied and the convergence rate of the resulting algorithm depends on the eigengap of $P_K(W)$. Maximizing this quantity for a fixed K leads to a common acceleration scheme known as Chebyshev acceleration [25, 26] and the choice

$$P_K(x) = 1 - \frac{T_K(c_2(1-x))}{T_K(c_2)}, \tag{16}$$

where $c_2 = \frac{1+\gamma}{1-\gamma}$ and T_K are the Chebyshev polynomials [25] defined as $T_0(x) = 1$, $T_1(x) = x$, and, for all $k \geq 1$,

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x). \tag{17}$$

Finally, verifying that this particular choice of $P_K(W)$ is indeed a gossip matrix, and taking $K = \lfloor \frac{1}{\sqrt{\gamma}} \rfloor$ leads to Alg. (2) with an optimal convergence rate with respect to γ and κ_l.

Theorem 4. *The iterative scheme in Alg. (2) converges to $\Theta = \theta^* \mathbb{1}^\top$ where θ^* is the solution of Eq. (1). Furthermore, the time needed for this algorithm to reach any given precision $\varepsilon > 0$ is*

$$O\left(\sqrt{\kappa_l}\left(1+\frac{\tau}{\sqrt{\gamma}}\right)\ln\left(\frac{1}{\varepsilon}\right)\right). \tag{18}$$

The proof of Theorem 4 relies on standard properties of Chebyshev polynomials that imply that, for the particular choice of $K = \lfloor \frac{1}{\sqrt{\gamma}} \rfloor$, we have $\frac{1}{\sqrt{\gamma(P_K(W))}} \leq 2$. Hence, Theorem 3 applied to the gossip matrix $W' = P_K(W)$ gives the desired convergence rate. The complete proof is available as supplementary material.

4.3 Discussion and further developments

We now discuss several extensions to the proposed algorithms.

Algorithm 2 Multi-Step Dual Accelerated method

Input: number of iterations $T > 0$, gossip matrix $W \in \mathbb{R}^{n\times n}$, $c_1 = \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}$, $c_2 = \frac{1+\gamma}{1-\gamma}$, $c_3 = \frac{2}{(1+\gamma)\lambda_1(W)}$, $K = \left\lfloor \frac{1}{\sqrt{\gamma}} \right\rfloor$, $\eta = \frac{\alpha(1+c_1^{2K})}{(1+c_1^K)^2}$, $\mu = \frac{(1+c_1^K)\sqrt{\kappa_l}-1+c_1^K}{(1+c_1^K)\sqrt{\kappa_l}+1-c_1^K}$

Output: $\theta_{i,T}$, for $i = 1, ..., n$

$x_0 = 0$, $y_0 = 0$
for $t = 0$ to $T-1$ **do**
  $\theta_{i,t} = \nabla f_i^*(x_{i,t})$, for all $i = 1, ..., n$
  $y_{t+1} = x_t - \eta$ AcceleratedGossip(Θ_t,W,K)
  $x_{t+1} = (1+\mu)y_{t+1} - \mu y_t$
end for

procedure AcceleratedGossip(x,W,K)
$a_0 = 1$, $a_1 = c_2$
$x_0 = x$, $x_1 = c_2 x(I - c_3 W)$
for $k = 1$ to $K-1$ **do**
  $a_{k+1} = 2c_2 a_k - a_{k-1}$
  $x_{k+1} = 2c_2 x_k(I - c_3 W) - x_{k-1}$
end for
return $x_0 - \frac{x_K}{a_K}$
end procedure

1. **Computation of $\nabla f_i^*(x_{i,t})$:** In practice, it may be hard to apply the dual algorithm when conjugate functions are hard to compute. We now provide three potential solutions to this problem: (1) *warm starts* may be used for the optimization problem $\nabla f_i^*(x_{i,t}) = \operatorname{argmin}_\theta f_i(\theta) - x_{i,t}^\top \theta$ by starting from the previous iteration $\theta_{i,t-1}$. This will drastically reduce the number of steps required for convergence. (2) SSDA and MSDA can be extended to composite functions of the form $f_i(\theta) = g_i(B_i\theta) + c\|\theta\|_2^2$ for $B_i \in \mathbb{R}^{m_i \times d}$ and g_i smooth, and for which we know how to compute the proximal operator. This allows applications in machine learning such as logistic regression. See supplementary material for details. (3) Beyond the composite case, one can also add a small (well-chosen) quadratic term to the dual, and by applying accelerated gradient descent on the corresponding primal, get an algorithm that uses primal gradient computations and achieves almost the same guarantee as SSDA and MSDA (off by a $\log(\kappa_l/\gamma)$ factor).

2. **Local vs. global condition number:** MSDA and SSDA depend on the worst strong convexity of the local functions $\min_i \alpha_i$, which may be very small. A simple trick can be used to depend on the *average* strong convexity. Using the proxy functions $g_i(\theta) = f_i(\theta) - (\alpha_i - \bar{\alpha})\|\theta\|_2^2$ instead of f_i, where $\bar{\alpha} = \frac{1}{n}\sum_i \alpha_i$ is the average strong convexity, will improve the local condition number from $\kappa_l = \frac{\max_i \beta_i}{\min_i \alpha_i}$ to

$$\kappa_l' = \frac{\max_i \beta_i - \alpha_i}{\bar{\alpha}} - 1. \tag{19}$$

   Several algorithms, including EXTRA [4] and DIGing [7], have convergence rates that depend on the strong convexity of the global function α_g. However, their convergence rates are not optimal, and it is still an open question to know if a rate close to $O\left(\sqrt{\kappa_g}(1+\frac{\tau}{\sqrt{\gamma}})\ln(1/\varepsilon)\right)$

can be achieved with a decentralized algorithm.

3. **Asynchronous setting:** Accelerated stochastic gradient descent such as SVRG [27] or SAGA [28] can be used on the dual problem in Eq. (12) instead of accelerated gradient descent, in order to obtain an asynchronous algorithm with a linear convergence rate. The details and exact convergence rate of such an approach are left as future work.

5 Experiments

In this section, we compare our new algorithms, single-step dual accelerated (SSDA) descent and multi-step dual accelerated (MSDA) descent, to standard distributed optimization algorithms in two settings: least-squares regression and classification by logistic regression. Note that these experiments on simple generated datasets are made to assess the differences between existing state-of-the-art algorithms and the ones provided in Section 4, and do not address the practical implementation details nor the efficiency of the compared algorithms on real-world distributed platforms. The effect of latency, machine failures or variable communication time is thus left for future work.

5.1 Competitors and setup

We compare SSDA and MSDA to four state-of-the-art distributed algorithms that achieve linear convergence rates: distributed ADMM (D-ADMM) [5], EXTRA [4], a recent approach named DIGing [7], and the distributed version of accelerated gradient descent (DAGD) described in Section 3.2 and shown to be optimal among centralized algorithms. When available in the literature, we used the optimal parameters for each algorithm (see Theorem 2 by [5] for D-ADMM and Remark 3 by [4] for EXTRA). For the DIGing algorithm, the parameters provided by [7] are very conservative, and lead to a very slow convergence. We thus manually optimized the parameter for this algorithm. The experiments are simulated using a generated dataset consisting of $10,000$ samples randomly distributed to the nodes of a network of size 100. In order to assess the effect of the connectivity of the network, we ran each experiment on two networks: one 10×10 grid and an Erdös-Rényi random network with parameter $p = \frac{6}{100}$ (i.e. of average degree 6). The quality metric used in this section is be the maximum approximation error among the nodes of the network

$$e_t = \max_{i \in \mathcal{V}} \bar{f}(\theta_{i,t}) - \bar{f}(\theta^*), \tag{20}$$

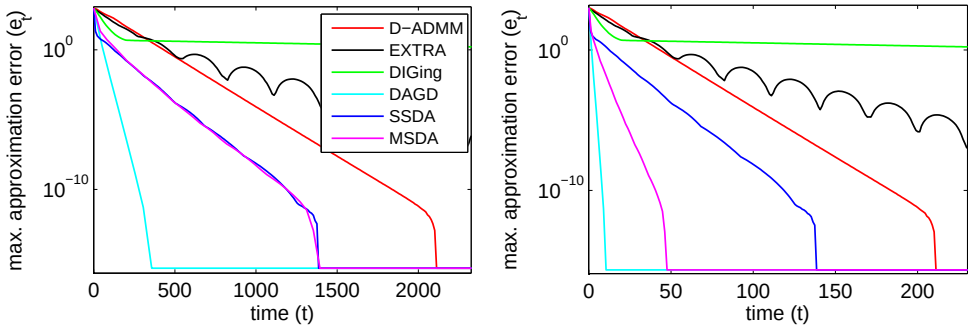
where θ^* is the optimal parameter of the optimization problem in Eq. (1).

5.2 Least-squares regression

The *regularized least-squares regression* problem consists in solving the optimization problem

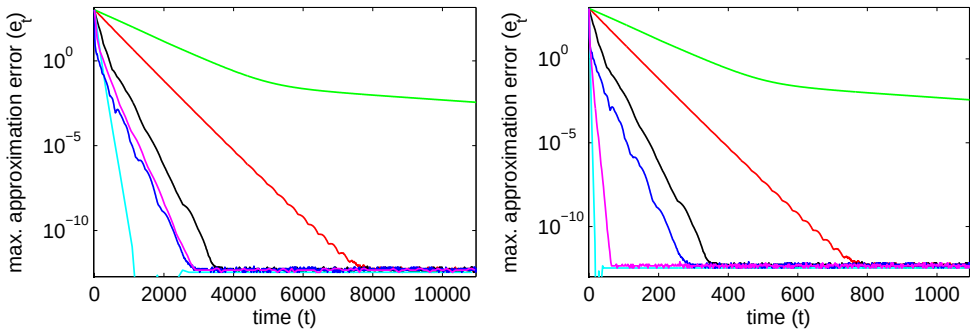
$$\min_{\theta \in \mathbb{R}^d} \frac{1}{m} \|y - X^\top \theta\|_2^2 + c\|\theta\|_2^2, \tag{21}$$

where $X \in \mathbb{R}^{d \times m}$ is a matrix containing the m data points, and $y \in \mathbb{R}^m$ is a vector containing the m associated values. The task is thus to minimize the empirical quadratic error between a function $y_i = g(X_i)$ of d variables and its linear regression $\hat{g}(X_i) = X_i^\top \theta$ on the original dataset (for



(a) high communication time: $\tau = 10$ (b) low communication time: $\tau = 0.1$

Figure 1: Maximum approximation error for least-squares regression on an Erdös-Rényi random network of average degree 6 ($n = 100$).

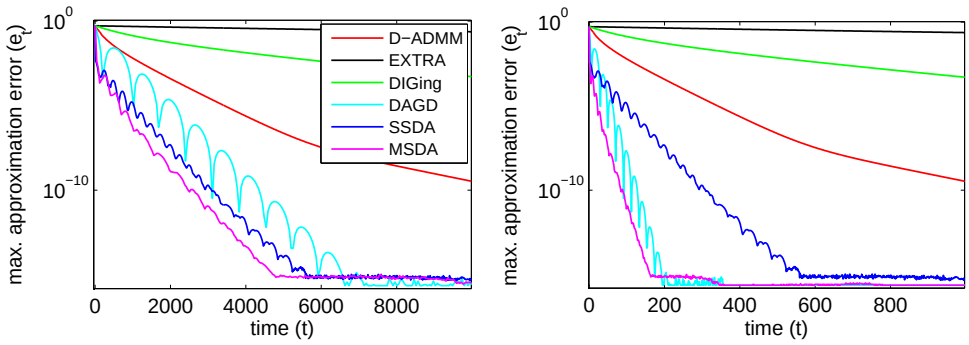


(a) high communication time: $\tau = 10$ (b) low communication time: $\tau = 0.1$

Figure 2: Maximum approximation error for least-squares regression on a 10×10 grid graph ($n = 100$).

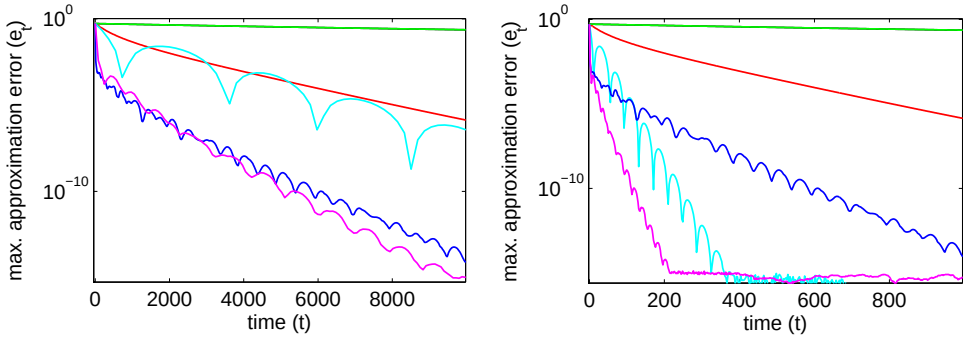
$i = 1, ..., m$), while smoothing the resulting approximation by adding a regularizer $c\|\theta\|_2^2$. For our experiments, we fixed $c = 0.1$, $d = 10$, and sampled $m = 10,000$ Gaussian random variables $X_i \sim \mathcal{N}(0, 1)$ of mean 0 and variance 1. The function to regress is then $y_i = X_i^\top \mathbb{1} + \cos(X_i^\top \mathbb{1}) + \xi_i$ where $\xi_i \sim \mathcal{N}(0, 1/4)$ is an i.i.d. Gaussian noise of variance $1/4$. These data points are then distributed randomly and evenly to the $n = 100$ nodes of the network. Note that the choice of function to regress y does not impact the Hessian of the objective function, and thus the convergence rate of the optimization algorithms.

Figure 1 and Figure 2 show the performance of the compared algorithms on two networks: a 10×10 grid graph and an Erdös-Rényi random graph of average degree 6. All algorithms are linearly convergent, although their convergence rates scale on several orders of magnitude. In all experiments, the centralized optimal algorithm DAGD has the best convergence rate, while MSDA has the best convergence rate among decentralized methods. When the communication time is smaller than the computation time ($\tau \gg 1$), performing several communication rounds per gradient iteration will improve the efficiency of the algorithm and MSDA substantially outperforms SSDA.



(a) high communication time: $\tau = 10$ (b) low communication time: $\tau = 0.1$

Figure 3: Maximum approximation error for logistic classification on an Erdös-Rényi random network of average degree 6 ($n = 100$).



(a) high communication time: $\tau = 10$ (b) low communication time: $\tau = 0.1$

Figure 4: Maximum approximation error for logistic classification on a 10×10 grid graph ($n = 100$).

5.3 Logistic classification

The *logistic classification* problem consists in solving the optimization problem

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \ln\left(1 + e^{-y_i \cdot X_i^\top \theta}\right) + c\|\theta\|_2^2, \tag{22}$$

where $X \in \mathbb{R}^{d \times m}$ is a matrix containing m data points, and $y \in \{-1, 1\}^m$ is a vector containing the m class assignments. The task is thus to classify a dataset by learning a linear classifier mapping data points X_i to their associated class $y_i \in \{-1, 1\}$. For our experiments, we fixed $c = 0.1$, $d = 10$, and sampled $m = 10,000$ data points, $5,000$ for the first class and $5,000$ for the second. Each data point $X_i \sim \mathcal{N}(y_i \mathbb{1}, 1)$ is a Gaussian random variable of mean $y_i \mathbb{1}$ and variance 1, where $y_i = 2\mathbb{1}\{i \leq 5,000\} - 1$ is the true class of X_i. These data points are then distributed randomly and evenly to the $n = 100$ nodes of the network.

Figure 3 and Figure 4 show the performance of the compared algorithms for logistic classification on two networks: a 10×10 grid graph and an Erdös-Rényi random graph of average degree 6. As for

least-squares regression, all algorithms are linearly convergent, and their convergence rates scale on several orders of magnitude. In this case, the centralized optimal algorithm DAGD is outperformed by MSDA, although the two convergence rates are relatively similar. Again, when the communication time is smaller than the computation time ($\tau \gg 1$), performing several communication rounds per gradient iteration will improve the efficiency of the algorithm and MSDA substantially outperforms SSDA. Note that, in Figure 4(a), D-ADMM requires 383 iterations to reach the same error obtained after only 10 iterations of SSDA, demonstrating a substantial improvement over state-of-the-art methods.

6 Conclusion

In this paper, we derived optimal convergence rates for strongly convex and smooth distributed optimization in two settings: centralized and decentralized communications in a network. For the decentralized setting, we introduced the *multi-step dual accelerated* (MSDA) algorithm with a provable optimal linear convergence rate, and showed its high efficiency compared to other state-of-the-art methods, including distributed ADMM and EXTRA. The simplicity of the approach makes the algorithm extremely flexible, and allows for future extensions, including time-varying networks and an analysis for non-strongly-convex functions. Finally, extending our complexity lower bounds to time delays, variable computational speeds of local systems, or machine failures would be a notable addition to this work.

A Detailed proofs

A.1 Complexity lower bounds

Proof of Theorem 1. This proof relies on splitting the function used by Nesterov to prove oracle complexities for strongly convex and smooth optimization [8, 22]. Let $\beta \geq \alpha > 0$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a graph and $A \subset \mathcal{V}$ a set of nodes of $\mathcal{G}$. For all $d > 0$, we denote as $A_d^c = \{v \in \mathcal{V} \ : \ d(A, v) \geq d\}$ the set of nodes at distance at least d from A, and let, for all $i \in \mathcal{V}$, $f_i^A : \ell_2 \to \mathbb{R}$ be the functions defined as:

$$f_i^A(\theta) = \begin{cases} \frac{\alpha}{2n}\|\theta\|_2^2 + \frac{\beta-\alpha}{8|A|}(\theta^\top M_1 \theta - \theta_1) & \text{if } i \in A \\ \frac{\alpha}{2n}\|\theta\|_2^2 + \frac{\beta-\alpha}{8|A_d^c|}\theta^\top M_2 \theta & \text{if } i \in A_d^c \\ \frac{\alpha}{2n}\|\theta\|_2^2 & \text{otherwise} \end{cases} \tag{23}$$

where $M_1 : \ell_2 \to \ell_2$ is the infinite block diagonal matrix with $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ on the diagonal, and $M_2 = \begin{pmatrix} 1 & 0 \\ 0 & M_1 \end{pmatrix}$. First, note that, since $0 \preceq M_1 + M_2 \preceq 4I$, $\bar{f}^A = \frac{1}{n}\sum_{i=1}^n f_i^A$ is α-strongly convex and β-smooth. Then, Theorem 1 is a direct consequence of the following lemma:

Lemma 1. *If $A_d^c \neq \emptyset$, then for any $t \geq 0$ and any black-box procedure one has, for all $i \in \{1, ..., n\}$,*

$$\bar{f}^A(\theta_{i,t}) - \bar{f}^A(\theta^*) \geq \frac{\alpha}{2}\left(\frac{\sqrt{\kappa_g} - 1}{\sqrt{\kappa_g} + 1}\right)^{2\left(1 + \frac{t}{1+d\tau}\right)} \|\theta_{i,0} - \theta^*\|^2, \tag{24}$$

where $\kappa_g = \beta/\alpha$.

Proof. This lemma relies on the fact that most of the coordinates of the vectors in the memory of any node will remain equal to 0. More precisely, let $k_{i,t} = \max\{k \in \mathbb{N} \,:\, \exists \theta \in \mathcal{M}_{i,t} \text{ s.t. } \theta_k \neq 0\}$ be the last non-zero coordinate of a vector in the memory of node i at time t. Then, under any black-box procedure, we have, for any local computation step,

$$k_{i,t+1} \leq \begin{cases} k_{i,t} + \mathbb{1}\{k_{i,t} \equiv 0 \bmod 2\} & \text{if } i \in A \\ k_{i,t} + \mathbb{1}\{k_{i,t} \equiv 1 \bmod 2\} & \text{if } i \in A_d^c \\ k_{i,t} & \text{otherwise} \end{cases}. \tag{25}$$

Indeed, local gradients can only increase even dimensions for nodes in A and odd dimensions for nodes in A_d^c. The same holds for gradients of the dual functions, since these have the same block structure as their convex conjugates. Thus, in order to reach the third coordinate, algorithms must first perform one local computation in A, then d communication steps in order for a node in A_d^c to have a non-zero second coordinate, and finally, one local computation in A_d^c. Accordingly, one must perform at least k local computation steps and $(k-1)d$ communication steps to achieve $k_{i,t} \geq k$ for at least one node $i \in \mathcal{V}$, and thus, for any $k \in \mathbb{N}^*$,

$$\forall t < 1 + (k-1)(1+d\tau), k_{i,t} \leq k-1. \tag{26}$$

This implies in particular:

$$\forall i \in \mathcal{V}, k_{i,t} \leq \left\lfloor \frac{t-1}{1+d\tau} \right\rfloor + 1 \leq \frac{t}{1+d\tau} + 1. \tag{27}$$

Furthermore, by definition of $k_{i,t}$, one has $\theta_{i,k} = 0$ for all $k > k_{i,t}$, and thus

$$\|\theta_{i,t} - \theta^*\|_2^2 \geq \sum_{k=k_{i,t}+1}^{+\infty} {\theta_k^*}^2. \tag{28}$$

and, since $\bar{f}^A$ is α-strongly convex,

$$\bar{f}^A(\theta_{i,t}) - \bar{f}^A(\theta^*) \geq \frac{\alpha}{2}\|\theta_{i,t} - \theta^*\|_2^2. \tag{29}$$

Finally, the solution of the global problem $\min_\theta \bar{f}^A(\theta)$ is $\theta_k^* = \left(\frac{\sqrt{\beta}-\sqrt{\alpha}}{\sqrt{\beta}+\sqrt{\alpha}}\right)^k$. Combining this result with Eqs. (27), (28) and (29) leads to the desired inequality. □

Using the previous lemma with $d = \Delta$ the diameter of $\mathcal{G}$ and $A = \{v\}$ one of the pair of nodes at distance Δ returns the desired result. □

Proof of Theorem 2. Let $\gamma_n = \frac{1-\cos(\frac{\pi}{n})}{1+\cos(\frac{\pi}{n})}$ be a decreasing sequence of positive numbers. Since $\gamma_2 = 1$ and $\lim_n \gamma_n = 0$, there exists $n \geq 2$ such that $\gamma_n \geq \gamma > \gamma_{n+1}$. The cases $n = 2$ and $n \geq 3$ are treated separately. If $n \geq 3$, let $\mathcal{G}$ be the linear graph of size n ordered from node v_1 to v_n, and weighted with $w_{i,i+1} = 1 - a\mathbb{1}\{i = 1\}$. Then, if $A = \{v_1, ..., v_{\lceil n/32 \rceil}\}$ and $d = (1 - 1/16)n - 1$, we have $|A_d^c| \geq |A|$ and Lemma 1 implies:

$$\bar{f}^A(\theta_{i,t}) - \bar{f}^A(\theta^*) \geq \frac{n\alpha}{2}\left(\frac{\sqrt{\kappa_g}-1}{\sqrt{\kappa_g}+1}\right)^{2\left(1+\frac{t}{1+d\tau}\right)}\|\theta_{i,0} - \theta^*\|^2. \tag{30}$$

A simple calculation gives $\kappa_l = 1 + (\kappa_g - 1)\frac{n}{2|A|}$, and thus $\kappa_g \geq \kappa_l/16$. Finally, if we take W_a as the Laplacian of the weighted graph $\mathcal{G}$, a simple calculation gives that, if $a = 0$, $\gamma(W_a) = \gamma_n$ and, if $a = 1$, the network is disconnected and $\gamma(W_a) = 0$. Thus, by continuity of the eigenvalues of a matrix, there exists a value $a \in [0, 1]$ such that $\gamma(W_a) = \gamma$. Finally, by definition of n, one has $\gamma > \gamma_{n+1} \geq \frac{2}{(n+1)^2}$, and $d \geq \frac{15}{16}(\sqrt{\frac{2}{\gamma}} - 1) - 1 \geq \frac{1}{5\sqrt{\gamma}}$ when $\gamma \leq \gamma_3 = \frac{1}{3}$.

For the case $n = 2$, we consider the totally connected network of 3 nodes, reweight only the edge (v_1, v_3) by $a \in [0, 1]$, and let W_a be its Laplacian matrix. If $a = 1$, then the network is totally connected and $\gamma(W_a) = 1$. If, on the contrary, $a = 0$, then the network is a linear graph and $\gamma(W_a) = \gamma_3$. Thus, there exists a value $a \in [0, 1]$ such that $\gamma(W_a) = \gamma$, and applying Lemma 1 with $A = \{v_1\}$ and $d = 1$ returns the desired result, since then $\kappa_g \geq 2\kappa_l/3$ and $d = 1 \geq \frac{1}{\sqrt{3\gamma}}$. □

A.2 Convergence rates of SSDA and MSDA

Proof of Theorem 3. Each step of the algorithm can be decomposed in first computing gradients, and then communicating these gradients across all neighborhoods. Thus, one step takes a time $1 + \tau$. Moreover, the Hessian of the dual function $F^*(\lambda\sqrt{W})$ is

$$(\sqrt{W} \otimes I_d)\nabla^2 F^*(\lambda\sqrt{W})(\sqrt{W} \otimes I_d), \tag{31}$$

where $\otimes$ is the Kronecker product and I_d is the identity matrix of size d. Also, note that, in Alg.(2), the current values x_t and y_t are always in the image of $\sqrt{W} \otimes I_d$ (i.e. the set of matrices x such that $x^\top \mathbb{1} = 0$). The condition number (in the image of $\sqrt{W} \otimes I_d$) can thus be upper bounded by $\frac{\kappa_l}{\gamma}$, and Nesterov's acceleration requires $\sqrt{\frac{\kappa_l}{\gamma}}$ steps to achieve any given precision [22]. □

Proof of Theorem 4. First, since $P_K(W)$ is a gossip matrix, Theorem 3 implies the convergence of Alg.(3). In order to simplify the analysis, we multiply W by $\frac{2}{(1+\gamma)\lambda_1(W)}$, so that the resulting gossip matrix has a spectrum in $[1 - c_2^{-1}, 1 + c_2^{-1}]$. Applying Theorem 6.2 in [25] with $\alpha = 1 - c_2^{-1}$, $\beta = 1 + c_2^{-1}$ and $\gamma = 0$ implies that the minimum

$$\min_{p \in \mathbb{P}_K, p(0)=0} \max_{x \in [1-c_2^{-1}, 1+c_2^{-1}]} |p(t) - 1| \tag{32}$$

is attained by $P_K(x) = 1 - \frac{T_K(c_2(1-x))}{T_K(c_2)}$. Finally, Corollary 6.3 of [25] leads to

$$\gamma(P_K(W)) \geq \frac{1 - 2\frac{c_1^K}{1+c_1^{2K}}}{1 + 2\frac{c_1^K}{1+c_1^{2K}}} = \left(\frac{1 - c_1^K}{1 + c_1^K}\right)^2, \tag{33}$$

where $c_1 = \frac{1-\sqrt{\gamma}}{1+\sqrt{\gamma}}$, and taking $K = \lfloor \frac{1}{\sqrt{\gamma}} \rfloor$ implies

$$\frac{1}{\sqrt{\gamma(P_K(W))}} \leq \frac{1 + c_1^{\frac{1}{\sqrt{\gamma}}+1}}{1 - c_1^{\frac{1}{\sqrt{\gamma}}+1}} \leq 2. \tag{34}$$

The time required to reach an $\varepsilon > 0$ precision using Alg.(3) is thus upper bounded by $O\left((1 + K\tau)\sqrt{\frac{\kappa_l}{\gamma(P_K(W))}}\ln(1/\varepsilon)\right) = O\left(\sqrt{\kappa_l}(1 + \frac{1}{\sqrt{\gamma}}\tau)\ln(1/\varepsilon)\right)$. □

B Composite problems for machine learning

When the local functions are of the form

$$f_i(\theta) = g_i(B_i\theta) + c\|\theta\|^2, \tag{35}$$

where $B_i \in \mathbb{R}^{m_i \times d}$ and g_i is smooth and has proximal operator which is easy to compute (and hence also g_i^*), an additional Lagrange multiplier ν can be used to make the Fenchel conjugate of g_i appear in the dual optimization problem. More specifically, from the primal problem of Eq. (12), one has, with $\rho > 0$ an arbitrary parameter:

$$\begin{aligned}
\inf_{\Theta\sqrt{W}=0} F(\Theta) &= \inf_{\Theta\sqrt{W}=0,\ \forall i, x_i = B_i\theta_i} \frac{1}{n}\sum_{i=1}^n g_i(x_i) + c\|\theta_i\|_2^2 \\
&= \inf_{\Theta} \sup_{\lambda,\nu} \frac{1}{n}\sum_{i=1}^n \left\{ \nu_i^\top B_i\theta_i - g_i^*(\nu_i) + c\|\theta_i\|_2^2 \right\} + \frac{\rho}{n}\operatorname{tr}(\lambda^\top \Theta\sqrt{W}) \\
&= \sup_{\nu \in \prod_{i=1}^n \mathbb{R}^{m_i},\ \lambda\in\mathbb{R}^{d\times n}} -\frac{1}{n}\sum_{i=1}^n g_i^*(\nu_i) - \frac{1}{4cn}\sum_{i=1}^n \|B_i^\top\nu_i + \rho\lambda\sqrt{W}_i\|_2^2.
\end{aligned}$$

To maximize the dual problem, we can use (accelerated) proximal gradient, with the updates:

$$\begin{aligned}
\nu_{i,t+1} &= \inf_{\nu\in\mathbb{R}^{m_i}} g_i^*(\nu) + \frac{1}{2\eta}\left\|\nu - \nu_{i,t} + \frac{\eta}{2c}B_i(B_i^\top\nu_{i,t} + \rho\lambda_t\sqrt{W}_i)\right\|_2^2 \\
\lambda_{t+1} &= \lambda_t - \eta\frac{\rho}{2cn}\sum_{i=1}^n (B_i^\top\nu_{i,t} + \rho\lambda_t\sqrt{W}_i)\sqrt{W}_i^\top.
\end{aligned}$$

We can rewrite all updates in terms of $z_t = \lambda_t\sqrt{W} \in \mathbb{R}^{d\times n}$, as

$$\begin{aligned}
\nu_{i,t+1} &= \inf_{\nu\in\mathbb{R}^{m_i}} g_i^*(\nu) + \frac{1}{2\eta}\left\|\nu - \nu_{i,t} + \frac{\eta}{2c}B_i(B_i^\top\nu_{i,t} + \rho z_{i,t})\right\|_2^2 \\
z_{t+1} &= z_t - \eta\frac{\rho}{2cn}\sum_{i=1}^n (B_i^\top\nu_{i,t} + \rho z_i)W_i^\top.
\end{aligned}$$

In order to compute the convergence rate of such an algorithm, if we assume that:

- each g_i is μ-smooth,
- the largest singular value of each B_i is less than M,

then we simply need to compute the condition number of the quadratic function

$$Q(\nu,\lambda) = \frac{1}{2\mu}\sum_{i=1}^n \|\nu_i\|_2^2 + \frac{1}{4c}\sum_{i=1}^n \|B_i^\top\nu_i + \rho\lambda\sqrt{W}_i\|_2^2.$$

With the choice $\rho^2 = \frac{1}{\lambda_{\max}(W)}\left(\frac{c}{\mu} + M^2\right)$, it is lower bounded by $\left(1 + \mu\frac{M^2}{c}\right)\frac{4}{\gamma}$, which is a natural upper bound on κ_l/γ. Thus this essentially leads to the same convergence rate than the non-composite case with the Nesterov and Chebyshev accelerations, i.e. $\sqrt{\kappa_l/\gamma}$.

The bound on the conditional number may be shown through the two inequalities:

$$\begin{aligned} Q(\nu,\lambda) &\leqslant \frac{1}{2\mu}\sum_{i=1}^{n}\|\nu_i\|^2+\frac{1}{2c}\sum_{i=1}^{n}\|\rho\lambda\sqrt{W}_i\|_2^2+\frac{1}{2c}\sum_{i=1}^{n}\|B_i^\top\nu_i\|_2^2, \\ Q(\nu,\lambda) &\geqslant \frac{1}{2\mu}\sum_{i=1}^{n}\|\nu_i\|^2+\frac{1}{1+\eta}\frac{1}{4c}\sum_{i=1}^{n}\|\rho\lambda\sqrt{W}_i\|_2^2-\frac{1}{\eta}\frac{1}{4c}\sum_{i=1}^{n}\|B_i^\top\nu_i\|_2^2, \end{aligned}$$

with $\eta = M^2\mu/c$.

References

[1] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[3] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

[4] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[5] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

[6] Dušan Jakovetić, José MF Moura, and Joao Xavier. Linear convergence rate of a class of distributed augmented lagrangian algorithms. *IEEE Transactions on Automatic Control*, 60(4):922–936, 2015.

[7] A. Nedich, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *ArXiv e-prints*, 2016.

[8] Yurii Nesterov. *Introductory lectures on convex optimization : a basic course*. Kluwer Academic Publishers, 2004.

[9] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2508–2530, 2006.

[10] Dušan Jakovetić, Joao Xavier, and José MF Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.

[11] Aryan Mokhtari and Alejandro Ribeiro. DSA: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

[12] Ermin Wei and Asuman Ozdaglar. Distributed alternating direction method of multipliers. In *51st Annual Conference on Decision and Control (CDC)*, pages 5445–5450. IEEE, 2012.

[13] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro. A decentralized second-order method with exact linear convergence rate for consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):507–522, 2016.

[14] R. Tutunov, H. B. Ammar, and A. Jadbabaie. A distributed newton method for large scale consensus optimization. *ArXiv e-prints*, 2016.

[15] Yossi Arjevani and Ohad Shamir. On the iteration complexity of oblivious first-order optimization algorithms. In *33nd International Conference on Machine Learning*, pages 908–916, 2016.

[16] Yossi Arjevani and Ohad Shamir. Dimension-free iteration complexity of finite sum optimization problems. In *Advances in Neural Information Processing Systems 29*, pages 3540–3548, 2016.

[17] Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In *Advances in Neural Information Processing Systems 27*, pages 163–171, 2014.

[18] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems 28*, pages 1756–1764, 2015.

[19] Ohad Shamir and Nathan Srebro. Distributed stochastic optimization and learning. In *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 850–857. IEEE, 2014.

[20] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[21] Stephen Boyd, Persi Diaconis, Pablo Parrilo, and Lin Xiao. Fastest mixing markov chain on graphs with symmetries. *SIAM Journal on Optimization*, 20(2):792–819, 2009.

[22] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[23] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation : numerical methods*. Prentice-Hall International, 1989.

[24] N. Alon and V. D. Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, series B*, 38:73–88, 1985.

[25] W. Auzinger. *Iterative Solution of Large Linear Systems*. Lecture notes, TU Wien, 2011.

[26] M. Arioli and J. Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3):591–608, 2014.

[27] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323, 2013.

[28] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654, 2014.